2

AD-A211 883

VLSI Memo No. 89-524
March 1989

Area-Universal Networks

Ronald I. Greenberg

Abstract

An area-universal network is one which can efficiently simulate any other network of comparable area. This paper extends previous results on area-universal networks in several ways. First, it considers the size (amount of attached memory) of processors comprising the networks being compared. It shows that an appropriate universal network of area $\Theta(A)$ built from processors of size $\lg A$ requires only $O(\lg^2 A)$ slowdown *in bit-times* to simulate any network of area A, without any restriction on processor size or number of processors in the competing network. Furthermore, the universal network can be designed so that any message traversing a path of length d in the competing network need follow a path of only $O(d + \lg A)$ length in the universal network. Thus, the results are almost entirely insensitive to removal of the unit wire delay assumption used in previous work. This paper also derives upper bounds on the slowdown required by a universal network to simulate a network of larger area and shows that all of the simulation results are valid even without the usual assumption that computation and communication of the competing network proceed in separate phases.

89 9 01 032

Acknowledgements

This research was supported in part by the Defense Advanced Research Projects Agency under contract N00014-87-K-0825, by the Office of Naval Research under contract N00014-86-K-0593, and by a Fannie and John Hertz Foundation Fellowship.

Author Information

Greenberg: Laboratory for Computer Science, MIT, Room NE43-340, Cambridge, MA 02139. (617) 253-5293.

Area-Universal Networks

Ronald I. Greenberg
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
rig@theory.lcs.mit.edu

(preliminary version)
March 1, 1989

Abstract

An area-universal network is one which can efficiently simulate any other network of comparable area. This paper extends previous results on area-universal networks in several ways. First, it considers the size (amount of attached memory) of processors comprising the networks being compared. It shows that an appropriate universal network of area $\Theta(A)$ built from processors of size $\lg A$ requires only $O(\lg^2 A)$ slowdown *in bit-times* to simulate any network of area A, without any restriction on processor size or number of processors in the competing network. Furthermore, the universal network can be designed so that any message traversing a path of length d in the competing network need follow a path of only $O(d + \lg A)$ *length in the universal network.* Thus, the results are almost entirely insensitive to removal of the unit wire delay assumption used in previous work. This paper also derives upper bounds on the slowdown required by a universal network to simulate a network of larger area and shows that all of the simulation results are valid even without the usual assumption that computation and communication of the competing network proceed in separate phases.

1 Introduction

This paper provides several advances in the search for the best way to make use of a fixed amount of physical space when building a general-purpose parallel computer. The focus on space consumed by both processors and interconnect represents an attempt to better measure real-world costs than to merely count the number of processors. The results of this paper are stated in terms of area required under standard two-dimensional VLSI modeling assumptions. The extension to three-dimensions is fairly straightforward using the ideas in [2].

The notion of a routing network which is universal for a given amount of physical space was introduced by Leiserson in [5]. That paper introduces a class of routing networks referred to as fat-trees and shows that an appropriate n-processor network from this class can simulate (off-line) any other routing network connecting the same processors and occupying the same volume, with only $O(\lg^4 n)$ factor degradation in the time required. A slight restriction on the number of processors in the competing network was required, because Leiserson's fat-trees used area slightly more than linear in the number of processors.

The approach to proving fat-trees universal is twofold. First it is shown that any competing network can be mapped to a fat-tree without placing too great a communications load on any of the communication

This research was supported in part by the Defense Advanced Research Projects Agency under Contract N00014-87-K-0825, by the Office of Naval Research under Contract N00014-86-K-0593, and by a Fannie and John Hertz Foundation Fellowship.

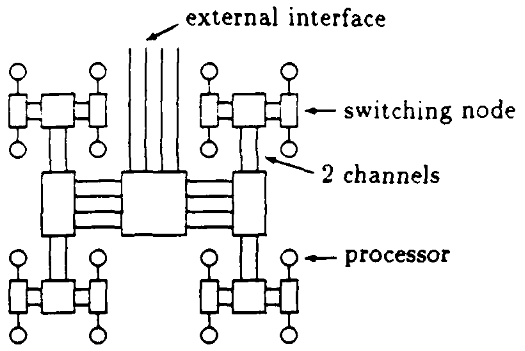

Figure 1: The organization of a fat-tree. Processors are located at the leaves, and the internal nodes contain concentrator switches. The channels between internal nodes consist of bundles of wires.

channels of the fat-tree. Then an efficient scheme for routing on fat-trees is provided to show that it can efficiently simulate the competing network.

In [3], the basic universality result for fat-trees was extended to the on-line case through demonstration of an on-line routing algorithm. Park [6] and Leighton, Maggs, and Rao [4] have obtained more efficient routing algorithms under different communications models. That is, they consider the effects of stronger switches or packet routing instead of circuit switching.

Leighton, Maggs, and Rao [4] use their routing algorithm to show that an appropriate universal network of area $\Theta(A)$ requires only $O(\lg^2 A)$ slowdown (in bit-times) to simulate any network of area A and the same type of processors, without any restriction on the number of processors. (Note that this result has been translated from $O(\lg A)$ slowdown in the word model. This paper expresses all times in bit-times.)

The focus in this paper is on the choice of universal network and the mapping of other networks to it. We obtain results about how good a network can be at facilitating this mapping and generally leave open the possibility of applying various routing approaches, but the results are occasionally specified concretely using the routing approach with the best known time bounds, that of [4]. Thus, it is possible to see the effect which would result from an improved routing algorithm as well as the current best known results.

This paper provides several improvements to previous universality results. For the simplest situation of comparing networks built out of the same (small) processors, the bounds demonstrated here match those of Leighton, Maggs, and Rao [4], though the network used here remains entirely within the fat-tree framework, which allows a more uniform routing strategy. This paper goes further by taking explicit note of the fact that fat-tree processors must be of size $\Omega(\lg A)$ in order to address all the other processors and that we may wish to consider networks with even larger processors. We make explicit the outcome of comparison between networks of larger and/or different processors and consider the question of best processor size for a universal network. Additionally, it is [illegible] that by superposing mesh connections on a fat-tree, assumptions of unit wire delay can be discarded. Also, global synchronization of competing networks is unnecessary.

The basic fat-tree network as introduced by Leiserson [5] is illustrated in Figure 1. Processors are located at the leaves of a complete binary tree, while switches are placed at the internal nodes. Each edge of the underlying tree corresponds to two *channels* of the fat-tree: one from parent to child, the other from child to parent. Unlike a normal tree which is "skinny all over," in a fat-tree, each channel consists of a bundle of wires. The number of wires in a channel c is called its capacity, denoted by cap(c).

It is the choice of channel capacities which is key to making fat-trees universal. The capacities must be large enough to efficiently support the communication of competing networks but small enough to keep the size of the fat-tree in check.

It is also important to note that fat-trees can be built using switches with a constant number of inputs and outputs [3] as in Figure 2. Since these switches are reminiscent of butterfly switches, such a fat-tree is

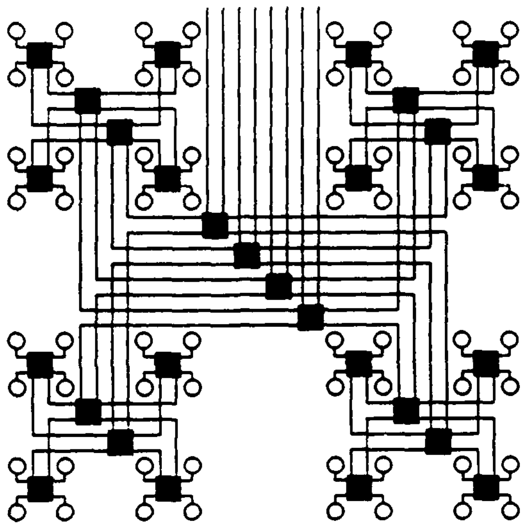

Figure 2: Another fat-tree design. The switches in this structure have constant size.

sometimes referred to as a butterfly fat-tree. There is still an underlying channel structure corresponding to the channels shown in Figure 1. By using two types of constant-size switches, it is possible to build butterfly fat-trees with essentially arbitrary channel capacities. The routing algorithms of [6] and [4] apply specifically to the butterfly fat-tree and are at least as efficient as the algorithms in [3]. (The former algorithms may require switches of size $\lg n$ but there is enough room for such switches in a "fold and squash" layout of a fat-tree [4].) Thus, we will henceforth use the term fat-tree to refer to a butterfly fat-tree.

The reason that we can largely separate the mapping and routing aspects of demonstrating universal networks is that the difficulty of routing a set M of messages between pairs of processors is well summarized by a measure we refer to as the *load factor*, $\lambda(M)$. First, let us define the *load* $\text{load}(M, c)$ of M on a channel c of a fat-tree to be the number of messages in M which must pass through c. Then the *load factor* of M on c is

$$\lambda(M, c) = \frac{\text{load}(M, c)}{\text{cap}(c)} ,$$

and the *load factor* of M on the entire fat-tree is

$$\lambda(M) = \max_{c} \lambda(M, c) .$$

The load factor λ is a lower bound on the time required to deliver the set of messages, and the routing algorithms which have been demonstrated for fat-trees obtain (high probability) upper bounds on the delivery time δ in the form

$$\delta(\lambda, n) = f(n)\lambda + g(n) ,$$

where f and g are small polylogarithmic functions of the number of processors, n.

Of particular interest is the case where n does not exceed 2^λ, implying that the delivery time is just a small polynomial function of λ. We will use δ with one argument to represent this case, i.e.,

$$\delta(\lambda) = \delta(\lambda, 2^\lambda) .$$

In [3], $\delta(\lambda, n) = \lambda \lg^3 n$ for a butterfly fat-tree, so that $\delta(\lg n)$ is $\lg^4 n$. With more powerful switches, Park [6] obtains $\delta(\lg n) = \lg^3 n$ deterministically. Leighton, Maggs, and Rao [4] obtain $\delta(\lambda, n) = \lambda + \lg n$ in the word model in a packet switched network. Since words, or messages, may need to be of length $\lg n$, this result should be thought of as $\delta(\lambda, n) = (\lambda + \lg n)\lg n$ and $\delta(\lg n) = \lg^2 n$ in order to maintain consistent measurement in terms of bit-times.

In this paper, we make a few important but reasonable assumptions relating to time and space. First of all, the technology being used determines a minimum feature size, which is our unit of space (though we initially maintain simplicity by using processor size as the measure of unit space). Second there is a fundamental lower bound on the time to switch a wire of unit length, which will be used as our basic unit of time. In VLSI technologies, such a bound is determined by the capacitance and resistance of a minimum-size transistor and the necessity of increasing capacitance if resistance is to be reduced. In fact, we will initially assume that this is the only source of delay and that transmission along any wire can be accomplished in unit time, but later we will relax this assumption. In any case, we assume that the number of bits which can leave a region of space in unit time is proportional to the perimeter of the region.

To make the task of simulating competing networks simpler, we will assume for now that the operation of the network is divided into separate phases of intraprocessor computation and interprocessor communication. Thus, to bound asymptotic simulation time, it will suffice to take the maximum of the overheads for simulation of the computation and simulation of the communication. Later, it is shown that this approach is valid, even if the competing network interleaves computation and communication in a more complicated fashion.

For convenience, we will use the following terminology throughout this paper:

> **Definition:** If, for any t, the operations performed by network A in time t can be performed by network B in time $t\Delta$, we say that network B can Δ-simulate network A.

The rest of this paper is organized as follows. Section 2 shows that an appropriate choice of channel capacities yields a universal fat-tree with processors packed as densely as possible (linear area for unit-size processors). Section 3 shows how stronger universality results may be obtained by generalizing to comparison of networks which may differ in processor size. Section 4 shows that the same results can virtually always be obtained even without the assumption of unit wire delay. Section 5 analyzes the ability of a universal network to simulate other networks which use more total area but the same processor area. Section 6 shows that the assumption of global synchronization of competing networks can be removed, and Section 7 contains concluding remarks.

2 A linear-size fat-tree

This section considers comparisons between networks built out of the same processors. It begins by constructing a fat-tree on unit-size processors which occupies area linear in the number of processors. With processors packed so densely, a very simple one-to-one mapping of a competing network's processors to those of the fat-tree ensures that message sets delivered in unit time by a competing network of area A have $O(\lg A)$ load factor in the fat-tree. Fat-tree routing mechanisms require processors of size at least logarithmic in the number of processors (in order to address all other processors), but by simply scaling the measure of area everywhere, the load factor result for unit-size processors leads to a valid simulation result for networks built out of the same processors of size $\Omega(\lg A)$. Nonetheless, we will explicitly introduce processor area since it yields an improved bound for very large processors and prepares the way for comparisons between networks built from different processors.

Specifying the universal fat-tree requires making an appropriate choice of the channel capacity function $c(p)$, which denotes the capacity of a fat-tree channel on top of a subtree of p processors. This section will show that the best construction may not be modular, i.e., the channel capacity on top of a subtree of the network may depend on the total size to which the network is to be built rather than depending just on the size of the subtree. It will be shown that modularity can be obtained using processors of size $\lg^2 A$ to build a fat-tree of area A, but we will see in Section 3 that this may increase the cost of simulating networks with processors of a different size if fat-tree routing strategies are developed which come closer to the lower bound on routing time.

Observe first that if we let $c(p) = \lceil \sqrt{p}/\lg A \rceil$, we can build a fat tree of A unit-size processors in area $\Theta(A)$. This can be derived by solving a recurrence relation for the sidelength $S(n)$ of a fat-tree on n processors in the H-tree layout style of Figure 1. We have

$$S(n) \quad = \quad 2S(n/4) + c(n) \ ,$$

with solution

$$S(n) = \sqrt{n}S(1) + \sum_{i=0}^{\log_4 n - 1} 2^i c(n/4^i) \ . \tag{1}$$

Substituting for $S(1) = 1$ and $c(n/4^i) \leq 1 + \sqrt{n/4^i}/\lg A$, we obtain

$$S(n) \quad \leq \quad 2\sqrt{n} + \sqrt{n}(\log_4 n)/\lg A$$

and $S(A) = \Theta(\sqrt{A})$.

Now we are prepared to present a result which strengthens the main universality result in [3] (translated to two dimensions) in that the restriction on the number of processors in the competing network is removed, but the proof here is actually more direct. Rather than using the ideas for balancing decomposition trees developed by Bhatt and Leighton [1] and Leiserson [5], we can consider a straightforward geometric mapping of competing networks to fat-trees. (This idea was actually introduced in [3] but only in the case of a fat-tree simulating networks of smaller area.)

Lemma 1 *With unit-sized processors, there exists a fat-tree, F, of area $\Theta(A)$ such that any message set delivered in unit time by a network R of area A induces a load factor of $O(\lg A)$ on F.*

Proof. We use the channel capacities $c(p) = \lceil \sqrt{p}/\lg A \rceil$ to build a fat-tree of $n = A$ processors, which we have shown requires area $\Theta(A)$. Then we can just recursively bisect R in the straightforward geometric fashion, cutting nonsquare pieces in the shorter direction, until we have A pieces. Since R can contain at most A processors, there can be at most one processor in each of the pieces of the recursive bisection, and these processors can be mapped to F so that the recursive bisection of R matches the obvious recursive bisection of F. Then the perimeter of a piece of R corresponding to a subtree of $n/2^l$ processors in F is $O(\sqrt{A}/2^{l/2})$. Thus, for a channel at level l of the decomposition, the load factor of messages generated in unit time is

$$\begin{aligned} \lambda &= O((\sqrt{A}/2^{l/2})/c(n/2^l)) \\ &= O((\sqrt{A}/2^{l/2})/(\sqrt{A}/(2^{l/2}\lg A))) \\ &= O(\lg A) \ . \blacksquare \end{aligned} \tag{2}$$

It is actually quite easy to generalize the above results to processors of size α. For simplicity, we assume that processors are square, i.e., $\sqrt{\alpha}$ by $\sqrt{\alpha}$. By building a fat-tree on $n = A/\alpha$ processors with channel capacity function $c(p) = \lceil \sqrt{p\alpha}/\lg(A/\alpha) \rceil$, we obtain area A and load factor $O(\lg(A/\alpha))$. This can be seen by simply making the correct substitutions into Equations 1 and 2. Thus, we have the following result:

Theorem 2 *When all processors are of size $\alpha = \Omega(\lg A)$, there exists a fat-tree, F, of area $\Theta(A)$ which can $O(\delta(\lg(A/\alpha)))$-simulate any network R of area A.*

Proof. We have seen that processors can be mapped one-to-one, and the message delivery time for each message set of the competing network is $\delta(O(\lg(A/\alpha))) = O(\delta(\lg(A/\alpha)))$. $\blacksquare$

Using the routing scheme of Leighton, Maggs, and Rao [4] yields the following corollary:

Corollary 3 *When all processors are of size $\alpha = \Omega(\lg A)$, there exists a fat-tree, F, of area $\Theta(A)$ which can $O(\lg^2(A/\alpha))$-simulate any network R of area A.* ■

This result is similar to a result given in [4] using a fat-tree with the lower levels replaced by meshes. By using the construction here, it is possible to instead retain a uniform routing strategy throughout all levels of the tree. The idea of adding mesh connections to a fat-tree is, however, a useful one when wire delay concerns are addressed, as we shall see in Section 4.

Two final notes are in order. First, when $\alpha = \Theta(\lg^2 n)$, we can achieve a modular design; we can use $c(p) = \lceil\sqrt{p}\rceil$, or rounding slightly differently, capacities which double at every other level as in Figure 2. Second, we can make the channel capacities slightly larger at the lower levels of the tree than was indicated above. (Such a change might help a routing algorithm to attain routing time closer to the load factor lower bound, as was the case for the algorithm in [3] for fat-trees in the style of Figure 1.) Specifically, if we increase channel capacities to $\lceil\sqrt{\alpha} + \sqrt{p\alpha}/\lg(A/\alpha)\rceil$, the asymptotic area of the fat-tree does not increase, as can be seen by substituting into Equation 1.

3 Different sized processors

In this section, we consider the possibility of comparing a universal network with processors of one size to other networks with processors of different size. In doing so, we must establish correspondences of single processors in one network with multiple processors in another network. We consider both many-to-one and one-to-many mappings of a competing networks' processors to those of a universal network. The combined observations show that a universal network designed to simulate networks of arbitrary processor size is best built with processors of size α in the range $\lg A \leq \alpha \leq \delta(\lg A)$.

In order to compare parallel machines within the framework of this paper, we must place some limitations on how we compare machines built from different processors. Rather than get involved in such issues as RISC vs. CISC or other detailed questions of best design for an individual processor, we assume that the processors of networks to be compared have the same instruction set and are equally well-engineered to provide the same operations at the same cost in time and space. The one difference in processors which remains under consideration is the amount of memory attached. For simplicity, we assume that the size of the memory does not affect the instruction execution time; incorporating a small cost for increasing memory size would cause little change to the results given here.

We consider in turn the two cases of the universal network having processors which are larger or smaller than the processors of the simulated network. In either case, let α_X represent the area of a processor in routing network X.

First, let us consider a competing network with processors smaller than those from which the universal network is to be built. In this case, we can map the competing network R to the universal fat-tree F as before except that the decomposition of R stops before we get down to individual processors. Instead, we map α_F/α_R processors of R to each processor of F. Ignoring for the moment, any communication to or from this block of processors, the computation performed by them in time t should be realizable in time $O(t\alpha_F/\alpha_R)$ on the processor of F. Thus, we can generalize Theorem 2:

Theorem 4 *For any $\alpha_F \geq \lg A$ and any $\alpha_R \leq \alpha_F$, there exists a fat-tree of area $\Theta(A)$ built out of processors of area α_F which can $O(\max\{\delta(\lg(A/\alpha_F)), \alpha_F/\alpha_R\})$-simulate any network of area A and processors of area α_R.* ■

Now, let us consider a competing network with processors larger than those from which the universal network is to be built. In this case, we decompose the competing network R down to individual processors as before, but we assign α_R/α_F processors of the fat-tree F to simulate each processor of R. We divide the memory of a processor of R among the corresponding α_R/α_F processors of F. Then as long as processors of F are large enough to address the α_R/α_F regions of memory, we can treat the flow of data to the different pieces of memory just as we would the communication among a set of processors being simulated. Thus, ignoring the communication to or from a processor of R, its computation can be simulated by the processors of F with $O(\delta(\lg(A/\alpha_F))$ slowdown. The same amount of overhead will suffice for simulating the interprocessor communication of R, yielding the following result:

Theorem 5 *For any $\alpha_F \geq \lg A$ and any $\alpha_R \geq \alpha_F$, there exists a fat-tree of area $\Theta(A)$ built out of processors of area α_F which can $O(\delta(\lg(A/\alpha_F)))$-simulate any network of area A and processors of area α_R.* ■

Since the overhead in Theorem 5, will never exceed $O(\delta(\lg A))$, we see that the overhead for a fat-tree competing against a network with larger processors is largely insensitive to the relative size of processors. Thus, if we wish to build a universal fat-tree to compete against networks of unknown processor size, it seems that we do best by making the fat-tree processors small enough that $\delta(\lg A)$ will always dominate α_F/α_R in comparisons against networks with smaller processors. Recalling that fat-tree routing mechanisms require that processors of F should be large enough to address $\Omega(A)$ processors, we are led to choose the processor size to satisfy $\lg A \leq \alpha_F \leq \delta(\lg A)$. Then we have the following corollaries to Theorems 4 and 5.

Corollary 6 *There exists a fat-tree of area $\Theta(A)$ which can $O(\delta(\lg A))$-simulate any network of area A and processors of arbitrary area.* ■

Corollary 7 *There exists a fat-tree of area $\Theta(A)$ which can $O(\lg^2 A)$-simulate any network of area A and processors of arbitrary area.* ■

4 Handling non-unit wire delay

In this section we consider the effect of dropping the unit wire delay assumption. The general graph layout framework developed by Bhatt and Leighton shows that there is enough room in our fat-tree layouts to build sufficiently large drivers for each wire to keep the wire delay constant in the capacitive model. This section shows that even if this constant switching time is not the dominant determiner of wire delay, the bounds on simulation time shown in earlier sections can almost always still be obtained. It provides a fat-tree structure augmented with mesh connections such that a routing path of length d in a competing network of area A corresponds to a path of length $O(d + \lg A)$ in the fat-tree. This will virtually always imply that asymptotic simulation overhead is no worse than in the unit wire delay case. Some of the ideas in this section were suggested by Charles Leiserson and Tom Cormen of MIT.

It should be noted that it is reasonable to assume wire delay to be better than linear in wire length since repeaters (extra switches) can always be used to reduce delay to linear. If, in fact, linear wire delay is required of any competing network, as would be the case if technology could be improved to the point where only speed of light limitations constrain the time to switch a length of wire (making the measure of unit time much smaller), the ultimate universal network is simply a mesh, but current technology remains far from this goal.

In order to obtain results independent of wire delay, we must begin with a regular layout of a fat-tree, that is one in which the components at any given level of the tree are regularly spaced throughout the layout. We can produce such a layout from a fat-tree with constant-size switches by using the "fold and squash" technique of Bhatt and Leighton [1, pp. 325–326] and Thompson [7, pp. 36–38]. Then with only a constant expansion in area, we can superpose a mesh on each level of components in the fat-tree. Such a transformation applied to the fat-tree in Figure 2 is illustrated in Figure 3 with inter-level connections omitted but switches labeled according to the number of levels up from the processors. In general, we must stop folding when we reach a level at which the channel capacities stop decreasing sufficiently (by a factor of 2 at every level in the underlying 4-ary tree), so we may be left with H-trees near the leaves.

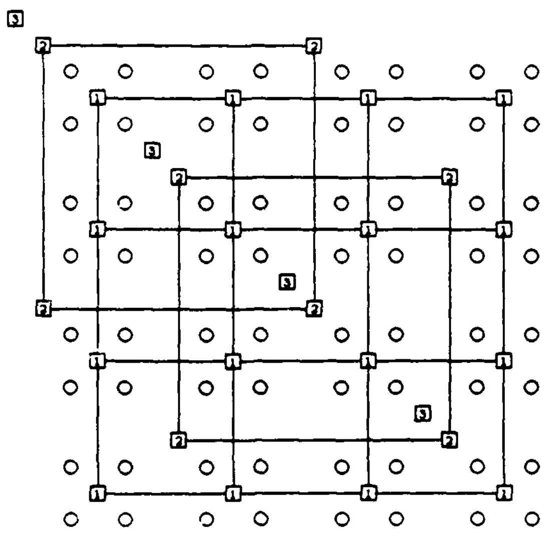

Figure 3: The arrangement of processors and switches and additional mesh connections used in the augmented fat-tree. Such a network allows good results to be obtained without the unit wire delay assumption.

Messages in the the augmented fat-tree are routed over the same paths as before, except that we allow each message to take one shortcut via one or two of the new mesh edges. More precisely the routing path is formed by going up tree edges until a switch is reached that is adjacent horizontally, vertically, or diagonally in the mesh at that level to a switch from which the destination can be reached by going down tree edges. As long as the mesh edges are of sufficiently large constant capacity, all of the existing routing algorithms will work as well as before; the shortcuts do not cause any extra messages to go through any of the tree edges, and the messages which go through any mesh edge are only those which would have gone up the tree from a constant number of nearby switches.

A key property of the layout in Figure 3 is that the wires connected to a switch l levels from the bottom of the tree are of length $O(2^l \lg A)$, where A is the area of the fat-tree. To see this, observe that the H-tree blocks near the leaves have sidelength $\lg A$, and the upper levels can be embedded in a tree of meshes graph of $O(\lg A)$ levels in such a way that each edge connected to a switch l levels up is mapped to at most $O(2^l)$ edges in the tree of meshes. The fold and squash layout of the tree of meshes given in [1] has maximum edge length $O(\lg A)$, and there is at most $\lg A$ extra length caused by the H-tree blocks in our layout. Thus, the length of an edge connected to a switch l levels up in the folded and squashed fat-tree is $O(2^l \lg A)$.

Now we can show that the mapping of competing networks to a universal fat-tree augmented with mesh connections does not stretch any wires by very much.

Theorem 8 *Let R be a network occupying a square of area A. Then, R can be mapped to an augmented fat-tree F of area $\Theta(A)$ so that any message following a path of length d in R will travel only $O(d + \lg A)$ distance in F.*

Proof. Observe first that if we use the straightforward mapping of R to F, processors separated by distance d in R are at most $\lceil d/\lg A \rceil$ H-tree blocks apart when mapped to F. Therefore, they are connected by a routing path which goes only $\lg(\lceil d/\lg A \rceil)$ levels up from the H-trees. This is true because four adjacent subtrees on $(\lceil d/\lg A \rceil)^2$ H-tree blocks must suffice to cover the processors of interest, implying that the routing path need only go up $\lg(\lceil d/\lg A \rceil)$ levels and use two mesh edges. Since any wire connected to a switch l levels up the tree is of length $O(2^l \lg A)$, the length of the routing path connecting processors at distance d in R is $O(d + \lg A)$. ■

Since wire delay can be held to linear, the above result virtually always insures that asymptotic simulation time does not degrade. It is easy to verify that this is so as long as the competing network must send messages over distances of $\Omega(\lg A)$. If the competing network in fact has processors communicating only over constant distance, then the simulation can be easily achieved as long as the augmented fat-tree includes also mesh connections of bandwidth as wide as possible between the processors. Then the competing network's communication can be achieved using only the mesh connections between processors, with only a constant number of messages per wire. Thus, the simulation overhead is only as great as the computation overhead involved in mapping several competing processors to one fat-tree processor plus $O(\lg A)$ time to transmit across the processor-to-processor mesh connections. For fat-tree processors of size $\lg A$, the total simulation overhead is therefore $O(\lg A)$. I will try to rigorously cover intermediate situations in the final paper, but it is clear that Corollaries 6 and 7 virtually always hold without the unit wire delay assumption.

5 Different sized networks

This section obtains upper bounds on the time required by a universal fat-tree to simulate networks that occupy more area but have the same amount of area devoted to processors. The reason for the latter restriction is that for any significant difference in memory, there are computations which can be performed in the larger amount of memory space but not in the smaller amount of memory space. Rather than placing restrictions on the type of computation, it is probably more meaningful to look at restrictions on the way that space is allocated. That is, if the larger network uses the same amount of processor area (including memory) and simply uses more interconnect area, then we can make meaningful comparisons between the networks. As would be expected, simulation difficulty increases as the area of the competing network does, but only up to a certain threshold beyond which extra area does not help the competition.

As we open up the issue of restricting the processor area used by competing networks, it may seem natural to ask about situations in which the competing network has less processor area than is allowed for the universal network. Indeed, we could have considered this question earlier when comparing networks of the same total area. But when the processor area of competing networks is so restricted, the best results are obtained by tailoring the universal network to the particular mix of processor and interconnect area, with the most difficult case occurring when the competing network has no less processor area than the universal network. Thus, the results given so far are worst case results for simulating networks of essentially the same total area. In this sense, these networks are the best known to build in a given area. Rather than digress to networks tailored to particular mixes of processors and interconnect, we now ask how well the networks discussed so far can do when they are actually matched against networks of larger area but with no greater processor area.

In what follows, we will let A_X represent the area of network X. Of necessity, however, we consider only competing networks in which the processor area does not exceed that of our universal fat-tree F. We are, of course, interested in the case where the competing network R has at least as much area as F, i.e., $A_R \geq A_F$. When $A_R \leq A_F$, we can simply invoke our earlier results. For simplicity, we will assume unit-size processors; the other variations discussed in the previous subsection are easily incorporated.

We use the same basic strategy as before for demonstrating universality results; that is, we recursively bisect the competing network and map appropriate pieces to the fat-tree processors. But when the competing network may have greater area than processor area, extra care is required to ensure that the decomposition is balanced; that is, when we bisect the area of the competing network, we must also bisect the competing processors (or pieces of processor area as in Section 3). Fortunately, we can invoke the general theory developed by Bhatt and Leighton [1] and, in a fashion that is cleaner for our purposes, by Leiserson [5]. (It is not desirable to use this approach when unnecessary due to a "loss of locality" in the mapping, which destroys the results on nonunit wire delay in Section 4.) These results tell us that since the competing network of area A_R has a $(\sqrt{A_R}, \sqrt{2})$ decomposition tree, it has a balanced $(\sqrt{A_R}, \sqrt{2})$ decomposition tree. Thus, when we match the bandwidth of the cuts in the decomposition to the channel capacities of our universal fat-tree, we find that the load factor for a set of messages delivered in time t by R is at most $O(t\sqrt{A_R/A_F}\lg A_F)$. This yields the following theorem:

Theorem 9 *A universal fat-tree of area $O(A_F)$ can $O(\delta(\sqrt{A_R/A_F}\lg A_F, A_F))$-simulate any network of area $A_R \geq A_F$ but processor area $O(A_F)$.* ■

When the area of the competing network is much larger than the area of the universal fat-tree, we can actually do bett.. than is suggested by Theorem 9. When A_R is $\Omega(A_F^2)$, the competing network is limited more by the restriction on processor area than by its total area. This is true because communication out of a piece of network R is limited not only by the perimeter of that piece but also by the perimeter of the processors in the piece. Thus at level l in the balanced decomposition of R, only $O(tA_F/2^l)$ messages can leave a corresponding piece of R in time t. This yields the following result:

Theorem 10 *A universal fat-tree of area $O(A_F)$ can $O(\delta(\sqrt{A_F}\lg A_F, A_F))$-simulate any network having processor area $O(A_F)$.* ■

6 Asynchronous networks

In this section, we show that the simulation results we have presented are valid even if the competing network does not have processors synchronized into global computation and communication phases, but rather has a complicated interleaving of intraprocessor computation and interprocessor communication. It is merely necessary that if the competing network performs intraprocessor computation and interprocessor communication in parallel, then the simulating network also has this capability. We can show that the approach of separately bounding the time required to simulate computation and the time required to simulate communication remains valid. The naive approach is to repeatedly simulate the delivery of messages received during a period of unit time in the competing network followed by simulation of the intraprocessor instructions performed during that period of time. But this ignores the fact that one processor may be simulating many others which perform indivisible instructions of differing duration. A variation of the naive approach yields the desired result, however.

Lemma 11 *Suppose that the set of messages whose delivery is completed by network A during any period of unit time can be delivered by network B in time f. Suppose also that for any time t, the intraprocessor computation fully performed by network A in time t can be performed by network B in time tg. Then the complete operation of network A over time t can be simulated by network B in time $t(f+g) = O(t\max\{f,g\})$.*

Sketch of proof. The routing network of network B performs t iterations of delivering the messages whose delivery is completed in the corresponding periods of unit time by network A, while the processors of B perform the intraprocessor instructions in order of completion time in A. Communication and computation proceed in parallel, but instructions and messages wait as necessary for the messages or instructions they depend on. ■

It should be noted that the proof given does not destroy the on-line nature of the simulation as long as instructions have predictable execution times.

7 Conclusion

This paper has shown that a fat-tree network can efficiently simulate any other network built in the same amount of area, without significant additional restrictions. It has further shown that under modest assumptions about differing processors, a parallel machine based on the fat-tree network can efficiently simulate any other parallel machine built in the same area, regardless of the relative processor sizes.

This paper has also given other important results. First, dropping the unit wire delay assumption does not significantly affect the simulation time required by universal networks. Second, given a universal fat-tree, we can obtain bounds on the time to simulate larger networks of the same total processor area. Unfortunately the latter result is not readily extended to the case of nonunit wire delay, due to the use of decomposition trees that are balanced. It is an open question whether or not this extension can be achieved. Perhaps it could be shown that there is a balanced decomposition tree which will not force nearby processors to be mapped too far from each other in the universal fat-tree.

The results given in this paper suggest that when building a universal network of area A, the best processor size to use is $\Theta(\lg A)$, or $\Theta(\lg^2 A)$ if modularity is more important than the ability to simulate networks of very small processors. The apparent lesson for the design of general-purpose parallel machines is that as total memory demands increase, most of the effort should go into expanding the number of processors rather than individual processor size.

A key open question is whether or not the routing algorithms for universal networks can be improved. It is possible that a better fat-tree routing algorithm could yield only $O(\lg A)$ slowdown in bit-times. Furthermore, it might be possible to take greater advantage of the mesh connections, which were introduced into the augmented fat-tree to eliminate potential wire delay problems. If the notion of load factor is generalized to arbitrary networks by considering arbitrary cuts of the network as opposed to fat-tree channels, then network mappings onto the augmented fat-tree yield constant load factor. It is possible that a routing scheme more clever than one relying almost exclusively on the tree connections could yield better message delivery times.

Acknowledgements

Thanks to Charles Leiserson of MIT for suggesting many of the areas of research covered in this paper and for providing extensive comments on earlier drafts. Thanks also to Tom Cormen and Bruce Maggs of MIT for helpful discussions.

References

[1] S. N. Bhatt and F. T. Leighton. A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences*, 28(2):300–343, Apr. 1984.

[2] R. I. Greenberg and C. E. Leiserson. A compact layout for the three-dimensional tree of meshes. *Applied Mathematics Letters*, 1(2):171–176, 1988.

[3] R. I. Greenberg and C. E. Leiserson. Randomized routing on fat-trees. In S. Micali, editor, *Randomness and Computation*. Volume 5 of *Advances in Computing Research*, JAI Press, 1989. To appear. Earlier versions available in MIT/LCS/TM-307 and 26th IEEE Symp. Found. of Comp. Sci. (1985), 241–249.

[4] T. Leighton, B. Maggs, and S. Rao. Universal packet routing algorithms. In *29th Annual Symposium on Foundations of Computer Science*, pages 256–269, IEEE Computer Society Press, 1988.

[5] C. E. Leiserson. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE Trans. Computers*, C-34(10):892–901, Oct. 1985.

[6] J. K [illegible]. A deterministic routing algorithm for the butterfly-fat-tree. 1989. Unpublished manuscript.

[7] C. [illegible] ompson. *A Complexity Theory for VLSI*. PhD thesis, Department of Computer Science, Carnegie-Mellon University, 1980.